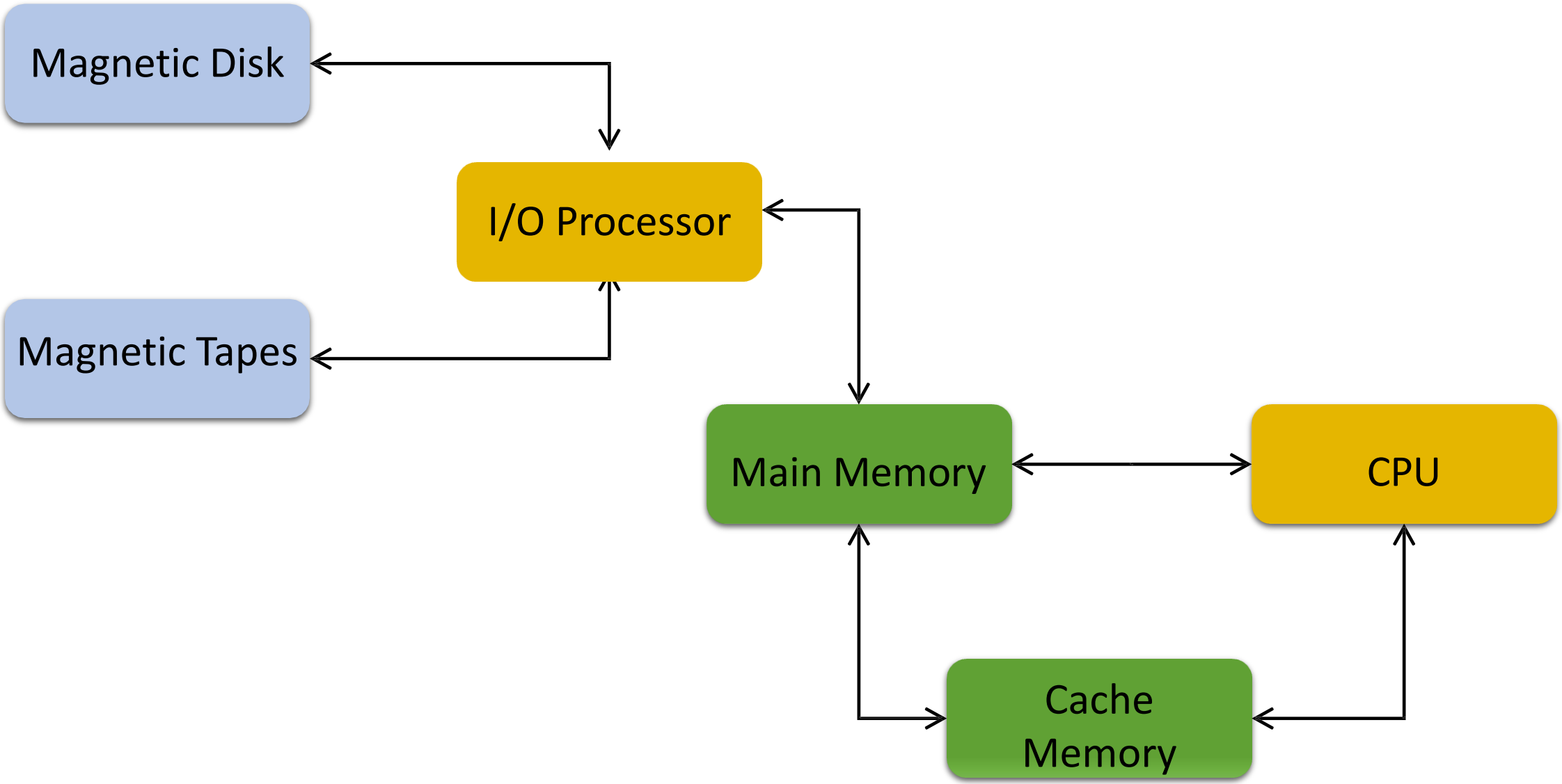
Unit 6: Memory Organization

Reference : Chapter 12 Computer System Architecture by Morris Mano

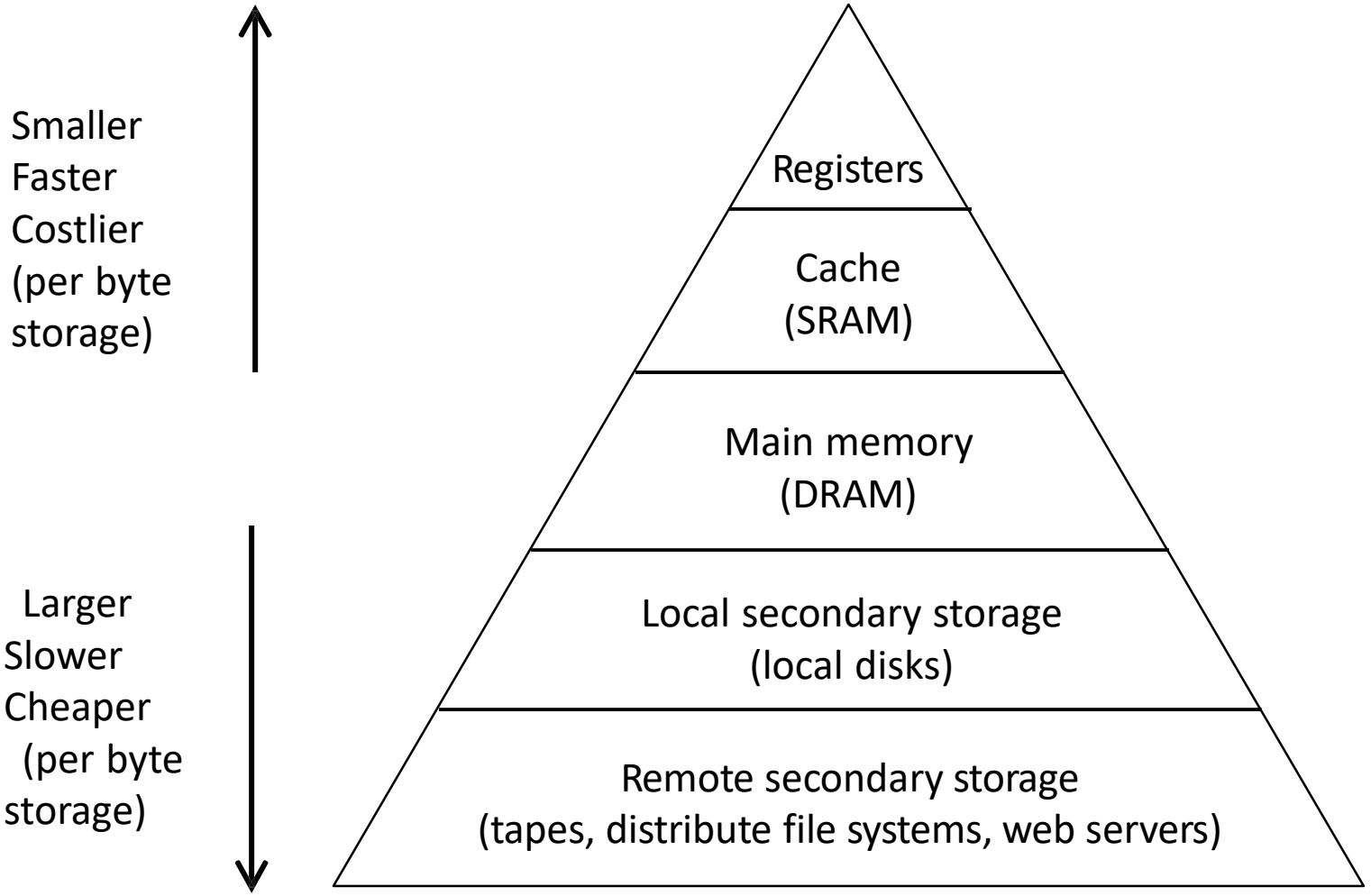
Memory Hierarchy

- What is Memory?
- Why different types of memory? (Main Memory, Auxiliary Memory)
    - Volatile, Large Storage, Speed
- Main Memory: memory unit that communicates directly with the CPU
    - RAM, ROM
- Auxiliary Memory: provide backup storage
    - Magnetic disk

Memory Organization and Relation



Memory Hierarchy



Main Memory

- Relatively large and fast memory used to store programs and data during the computer operation.

- RAM(Random Access Memory)
  - Static and Dynamic

- ROM(Read Only Memory)

RAM (Random Access Memory)

- Volatile
- Can perform Read and Write operation

1.SRAM (Static RAM)

- consist of internal flip-flops that store the binary information.

✓ Cache Memory

2.DRAM(Dynamic RAM)

- Stores the Binary information in the form of electric charges that are applied to capacitors
- Refreshing is required
- Reduced power consumption and Larger storage capacity

ROM (Read Only Memory)

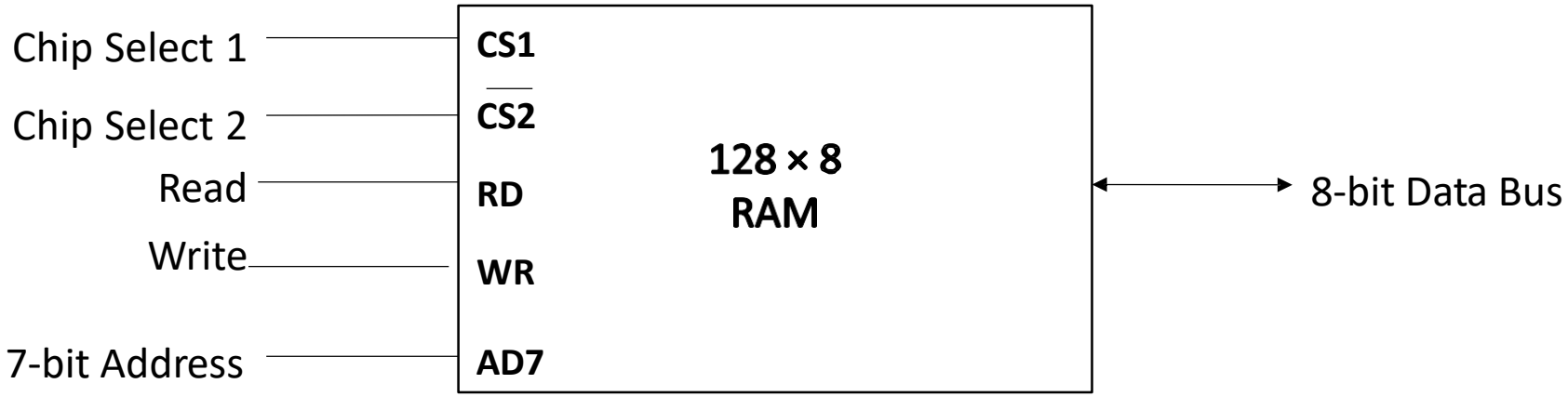
- Non volatile
- Can perform Read operation only
- Store programs and data that are permanently reside in the computer that do not change in value.
  - Store Bootstrap Loader

Bootstrap Loader

- It is a program whose function is to start the computer operating system when power is turned on.
- When power is turned on, the hardware of the computer sets the program counter to the first address of the bootstrap loader.
- The bootstrap program loads a portion of the operating system from disk to main memory and control is then transferred to the OS, which prepares the computer for general use.

RAM Chip

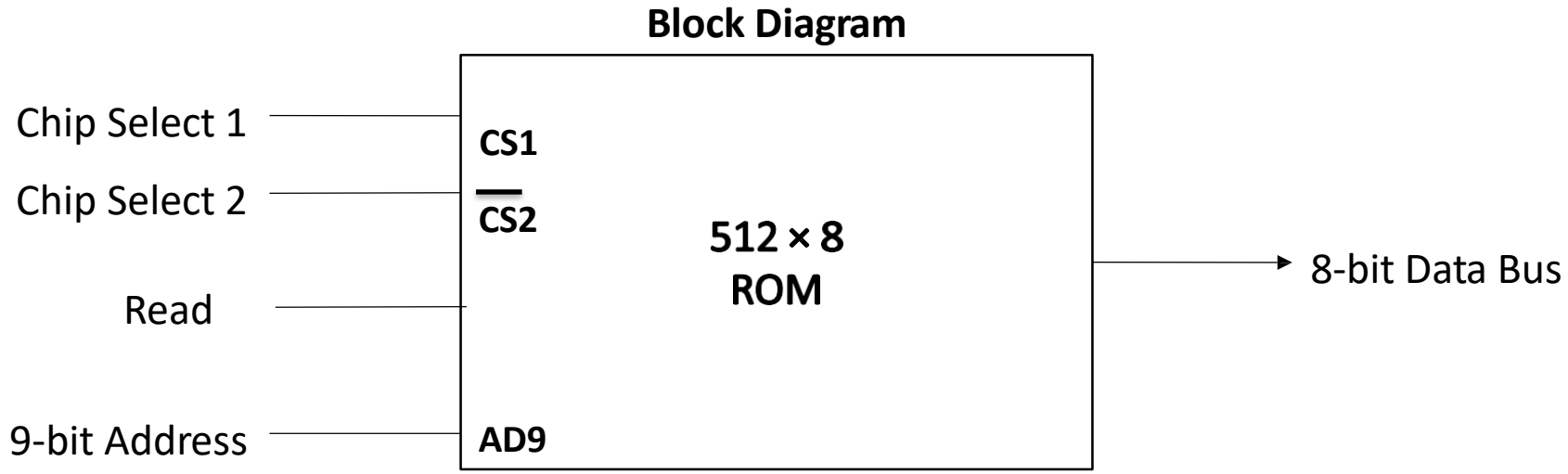
Block Diagram



Function Table

CS1	$\overline{CS2}$	RD	WR	Memory Function	State of the Data Bus
0	0	×	×	Inhibit	High Impedance
0	1	×	×	Inhibit	High Impedance
1	1	×	×	Inhibit	High Impedance
1	0	0	0	Inhibit	High Impedance
1	0	0	1	Write	Input data to RAM
1	0	1	0	Read	Output data from RAM

ROM Chip

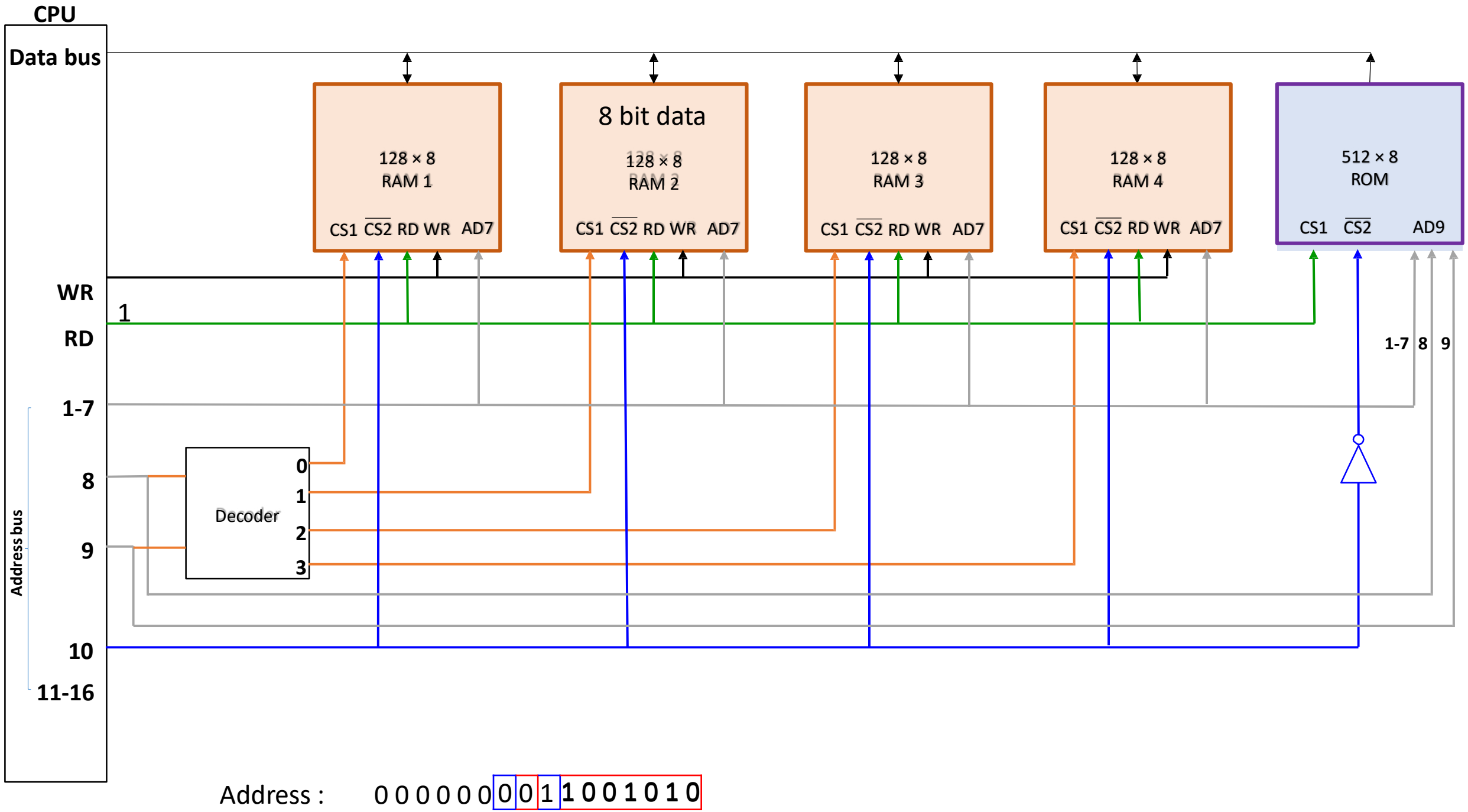

Block Diagram
Chip Select 1
Chip Select 2
Read
9-bit Address
CS1
CS2
AD9
512 × 8
ROM
8-bit Data Bus

Memory Connection to the CPU

- Require Memory: 512 Bytes of RAM and 512 Bytes of ROM

- Available chip
  - RAM: 128 × 8
  - ROM: 512 × 8

- No of RAM and ROM Chip required
  - 4 RAM Chip
  - 1 ROM Chip


CPU
Data bus
128 × 8
RAM 1
CS1 CS2 RD WR AD7
8 bit data
128 × 8
RAM 2
CS1 CS2 RD WR AD7
128 × 8
RAM 3
CS1 CS2 RD WR AD7
128 × 8
RAM 4
CS1 CS2 RD WR AD7
512 × 8
ROM
CS1 CS2 AD9
WR
1
RD
1-7
8
9
1-7
Address bus
Decoder
0
1
2
3
10
11-16
Address : 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0

Memory Address Map

Note: Bits 11 to 16 are all zero

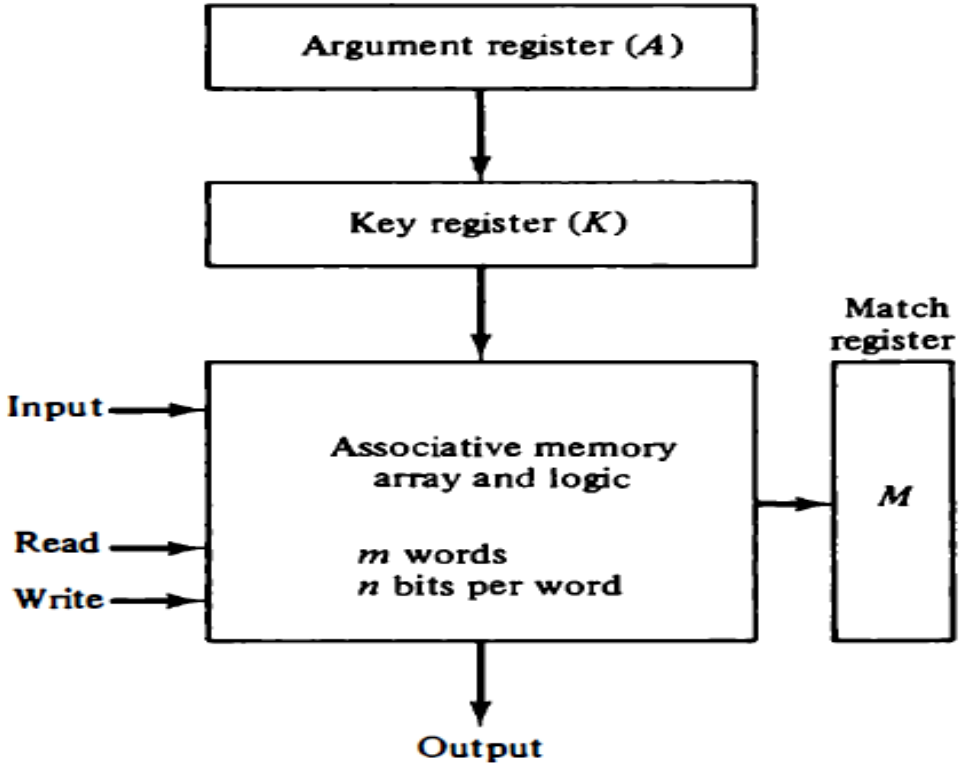
Component	Starting Address											Ending Address										
	Binary										Hexa Decimal	Binary										Hexa Decimal
	10	9	8	7	6	5	4	3	2	1		10	9	8	7	6	5	4	3	2	1	
RAM-1	0	0	0	0	0	0	0	0	0	0	0000	0	0	0	1	1	1	1	1	1	1	007F
RAM-2	0	0	1	0	0	0	0	0	0	0	0080	0	0	1	1	1	1	1	1	1	1	00FF
RAM-3	0	1	0	0	0	0	0	0	0	0	0100	0	1	0	1	1	1	1	1	1	1	017F
RAM-4	0	1	1	0	0	0	0	0	0	0	0180	0	1	1	1	1	1	1	1	1	1	01FF
ROM	1	0	0	0	0	0	0	0	0	0	0200	1	1	1	1	1	1	1	1	1	1	03FF

Associative Memory

- A memory unit accessed by content is called an **associative memory** or **content addressable memory(CAM)**.
- Time required to find an item stored in memory can be reduced.
- Can be accessed simultaneously.
- The memory is capable of finding an empty unused location to store the word.
- When a word is to be read from the memory, the content of the word, or part of the word, is specified.
- The memory locates all words which match the specified content and marks them for reading.
- Expensive

Associative Memory

Block Diagram of Associative Memory



Associative Memory

- **Argument Register (A):** It contains the word to be searched. It has n bits (One for each bit of the word)

- **Key Register (K):** This specifies which part of the argument words need to be compared with words in memory. If all bits are 1, the entire word should be compared. Otherwise only the bits having k-bits set to 1 will be compared

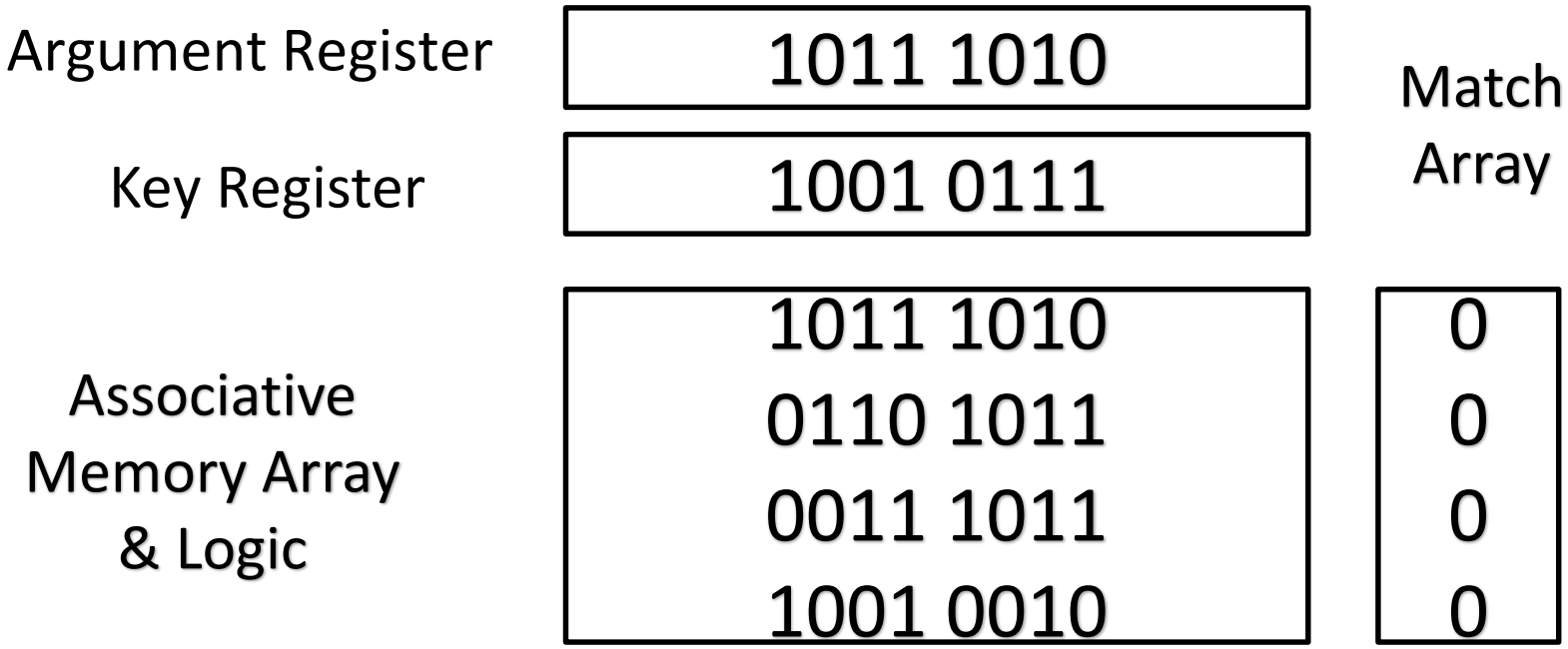
- **Associative Memory Array:** It contains the words which are to be compared with the Argument Word.

- **Match Register (M):** It has m bits, one bit corresponding to each word in the memory array. After the matching process, the bits corresponding to matching words in match register are set to 1.

Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

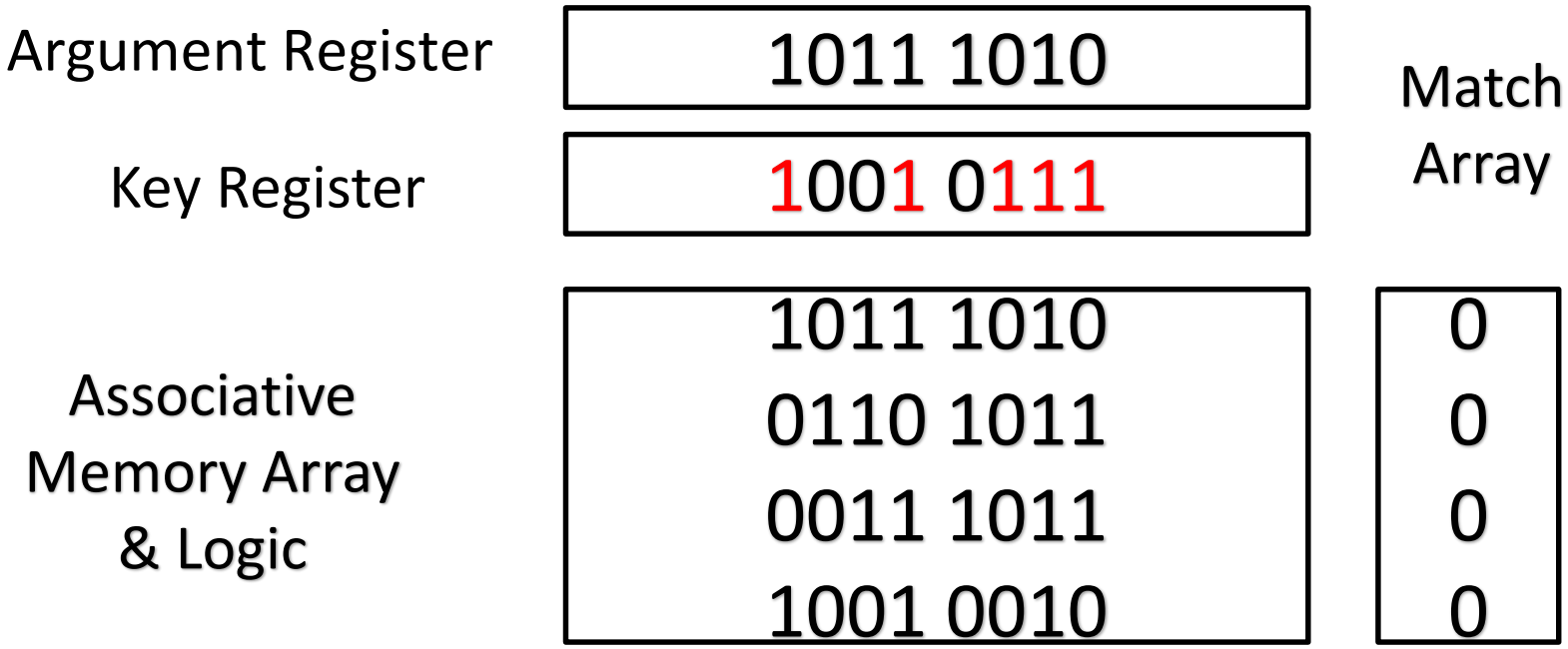
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

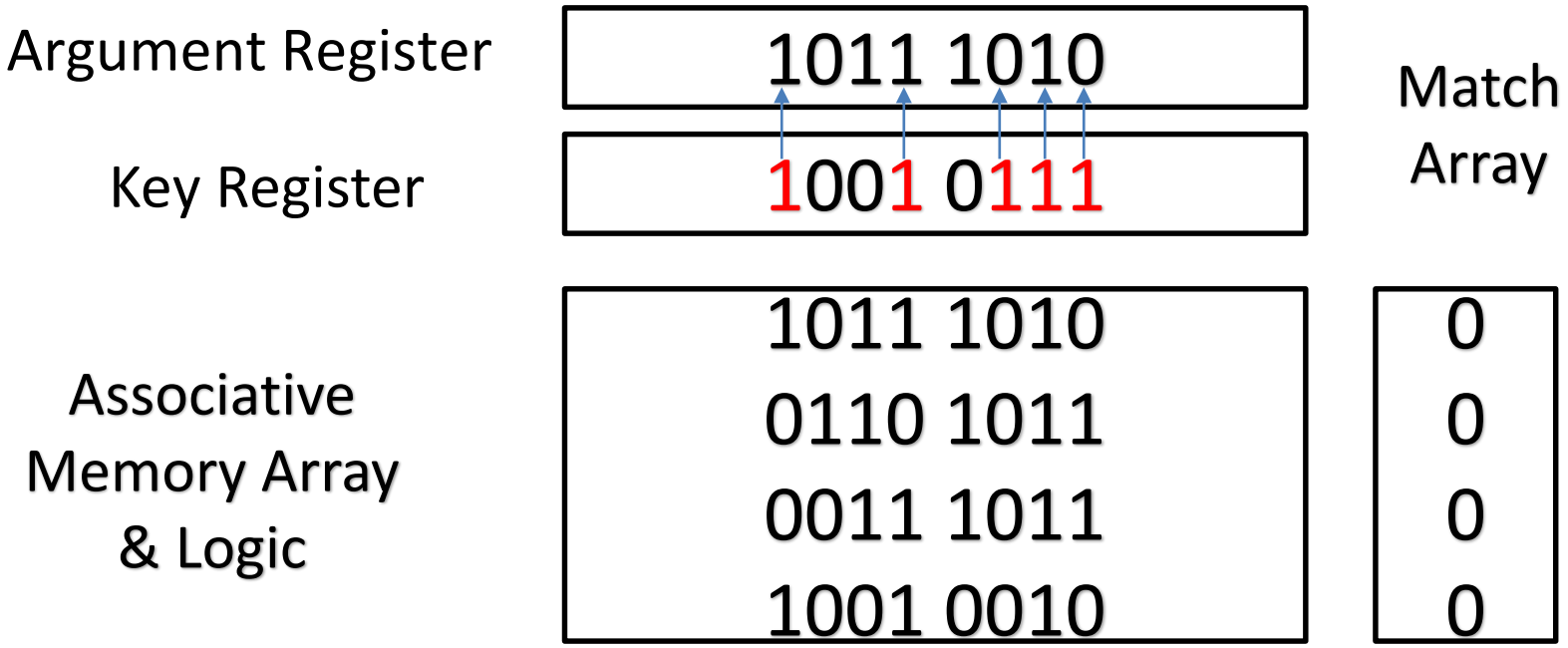
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

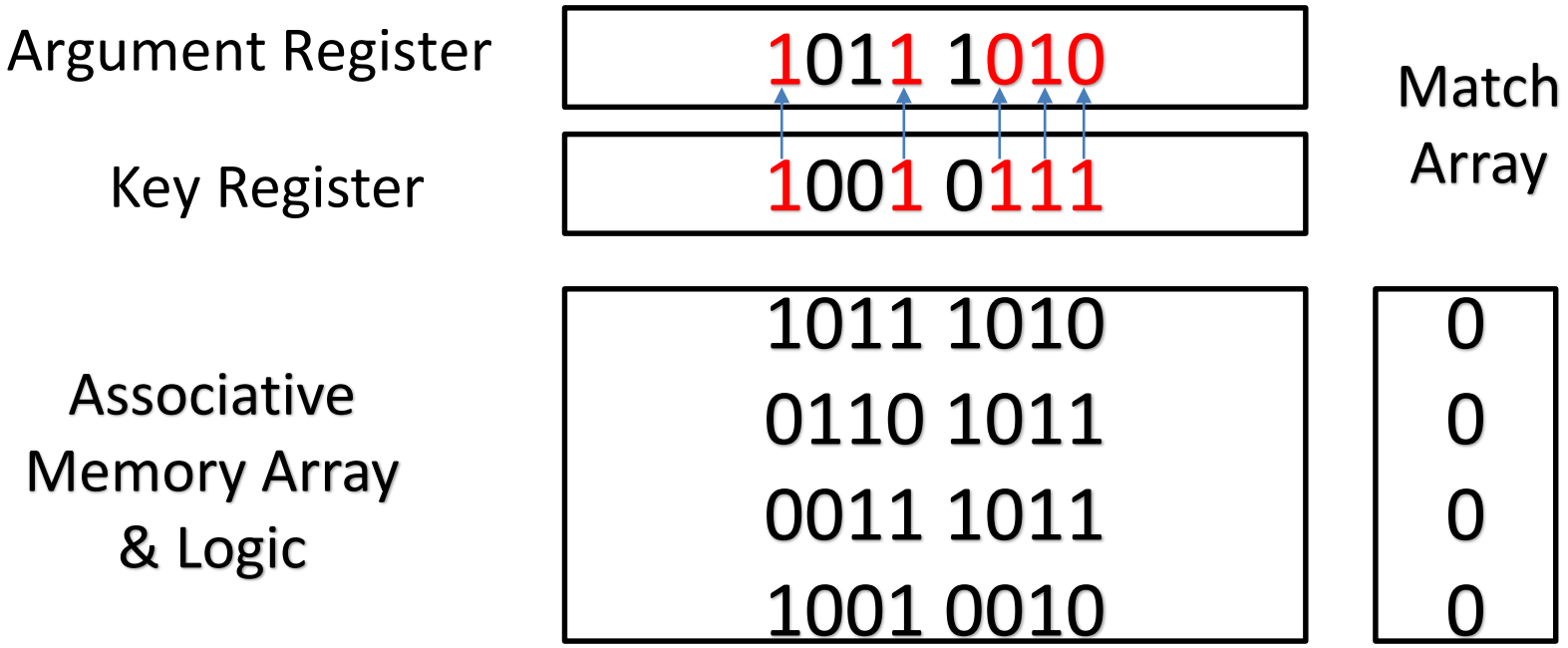
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

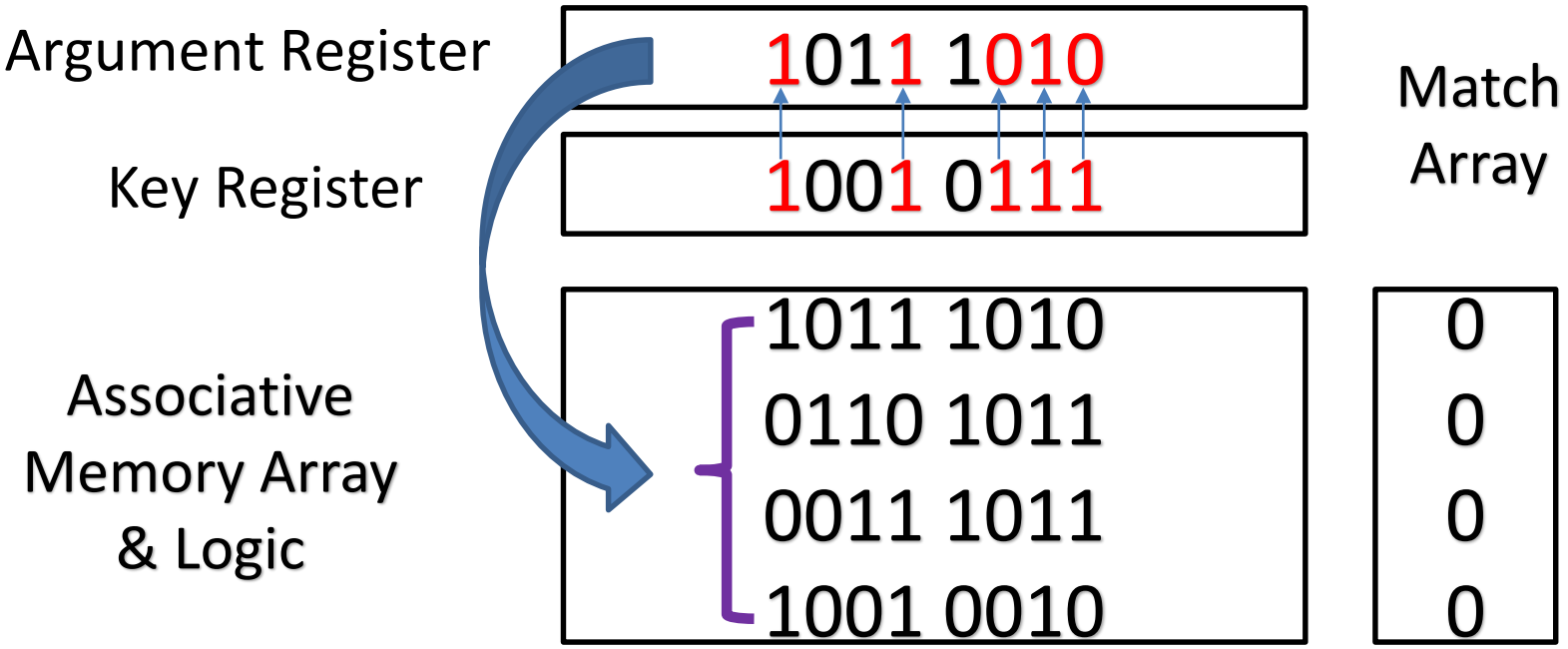
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

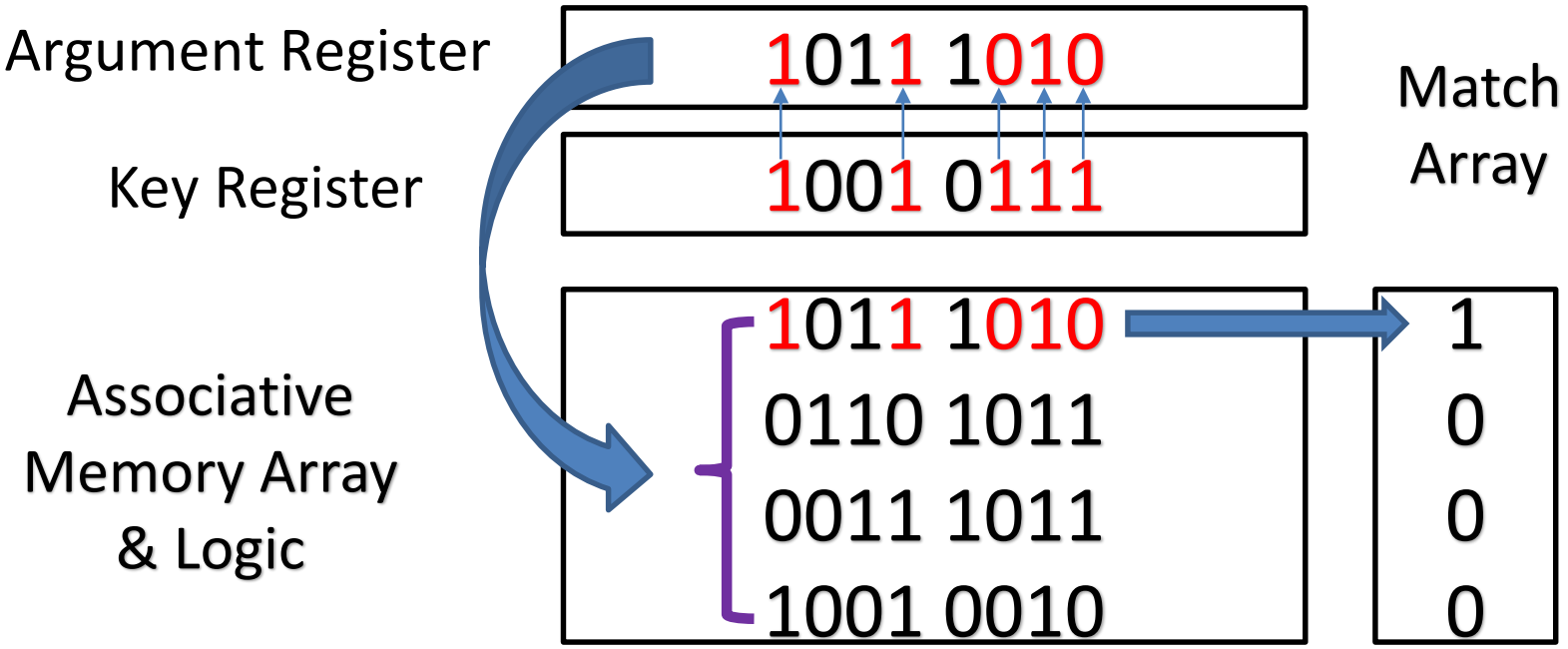
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

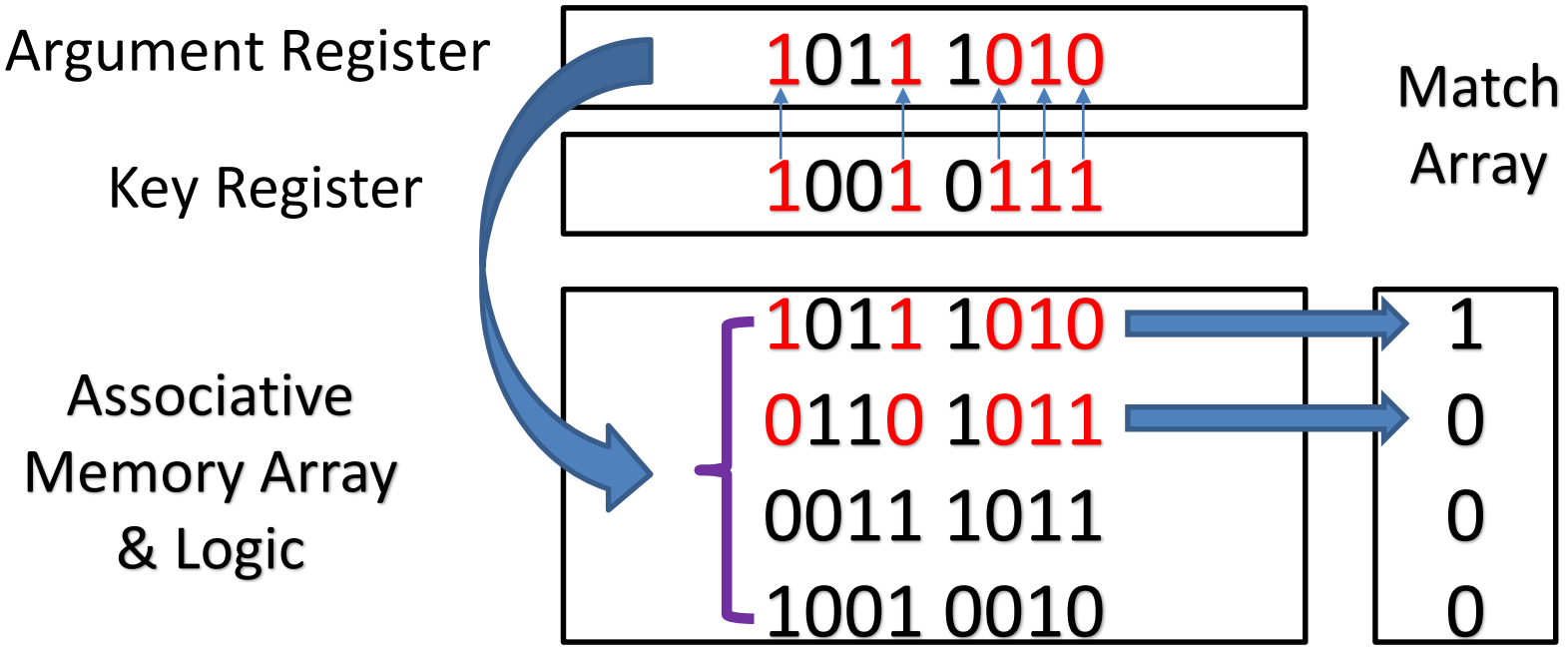
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

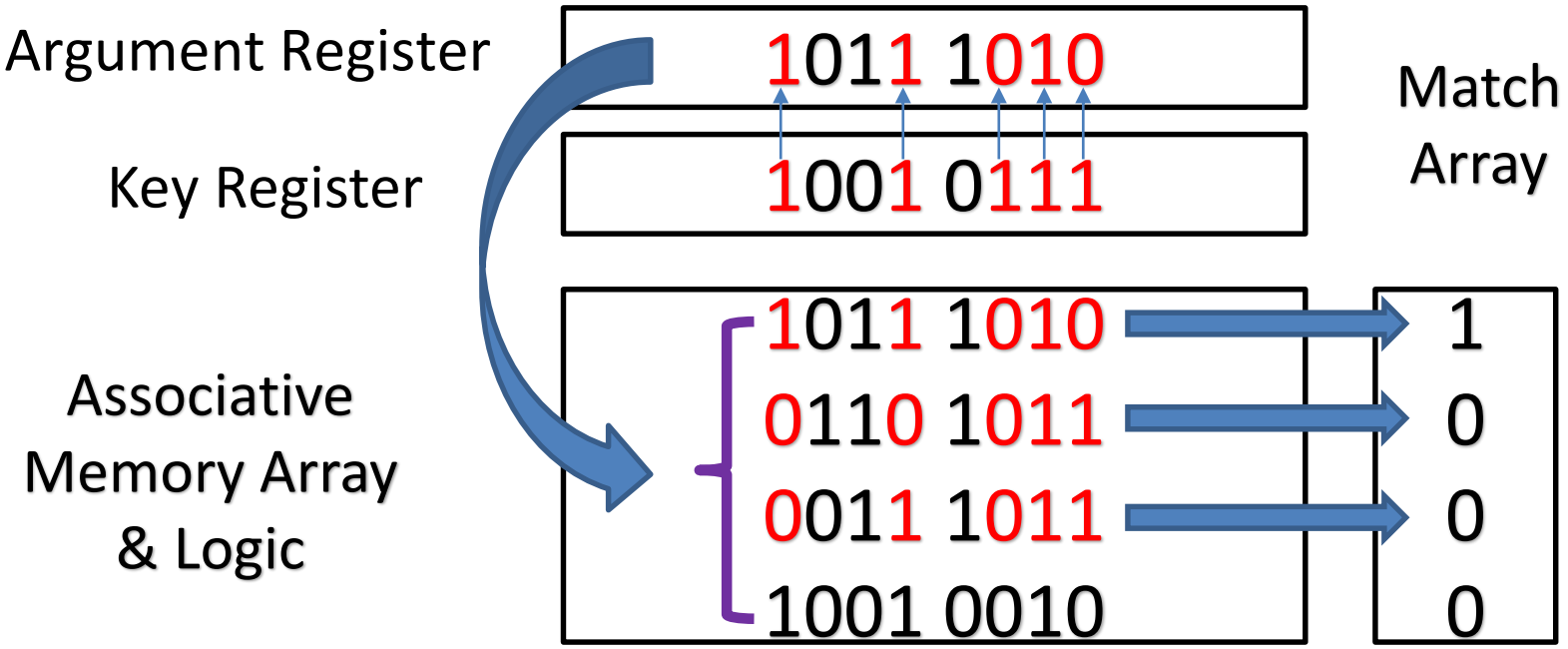
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

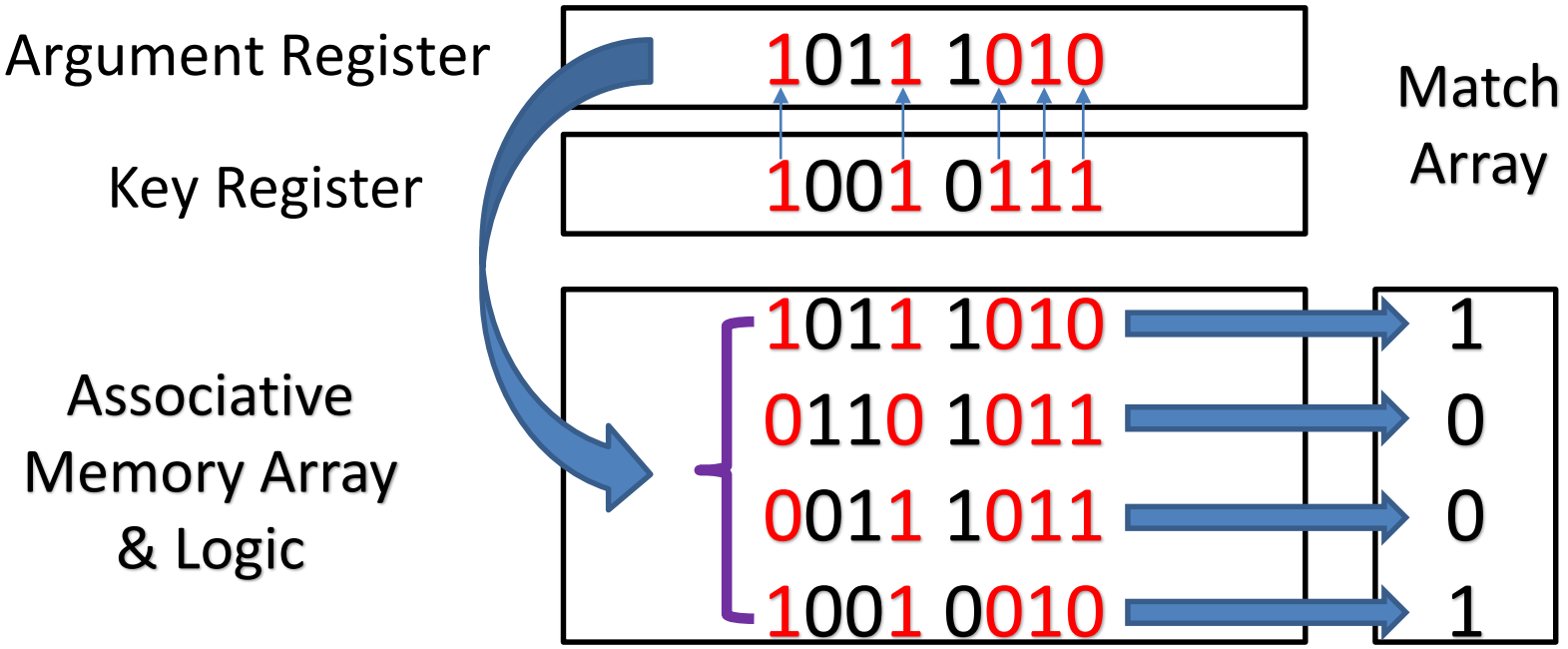
Example:



Associative Memory

- Reading is accomplished by sequential access in memory for those words whose bits are set.

Example:



Cache Memory

- **Locality of reference**: few localized areas of memory are repeatedly referred.

- If the active program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing total execution time of the program. Such a fast small memory is referred to as a **cache memory**.

- Less access time than main memory.

- What happens when CPU access memory?
  - CPU-----> Cache Memory ----->Main Memory

Cache Memory

- **Hit Ratio:** The performance of cache memory is frequently measured in terms of a quantity called Hit Ratio

- When the CPU refers the memory and finds the word in cache, it is said to produce a **hit**.

- If the word is not found in cache, it is in main memory and it count as a **miss**.

- **The ratio of the numbers of the hit is divided by the total CPU references to memory is the hit ratio**.

Performance of Cache Memory

- Hit ratio = $\frac{Number\ of\ Hits}{Total\ CPU\ refernce\ to\ memory(Hits+Misses)}$

- T_C = Cache Memory Access Time
- T_m = Main Memory Access Time
- T_{avg}= Average Memory Access Time
- H = Hit ratio

- $T_{avg} = H \times T_C + (1 - H) \times (T_C + T_m)$

Addressing Mapping

- **Mapping:** The transformation of data from main memory to cache memory is referred to do as a mapping process

- Decide which block of main memory has to be placed where in the cache memory.

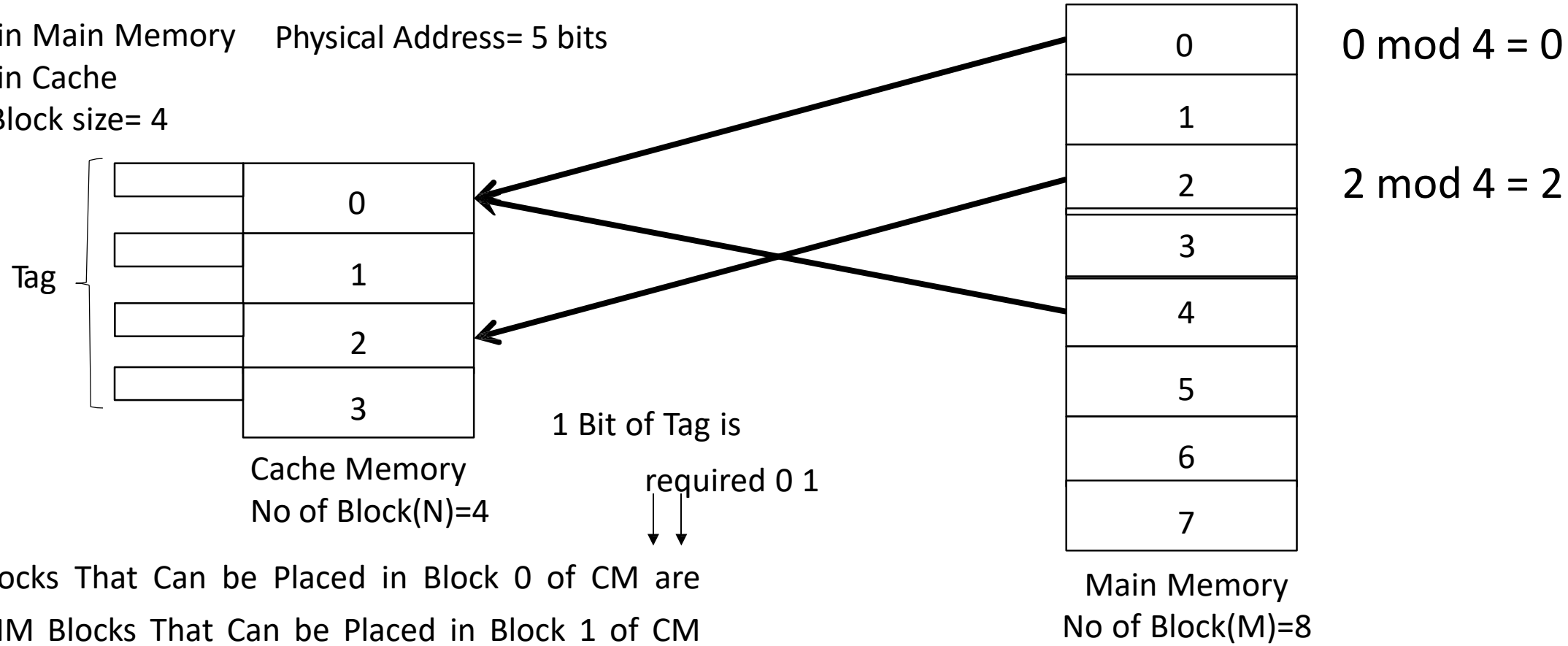
- Three types of Mapping process:
  1. Direct Mapping
  2. Associative Mapping
  3. Set – Associative Mapping

Direct Mapping

K^{th} Block of main memory is placed in $(K^{th} \bmod N)^{th}$ block of cache memory.

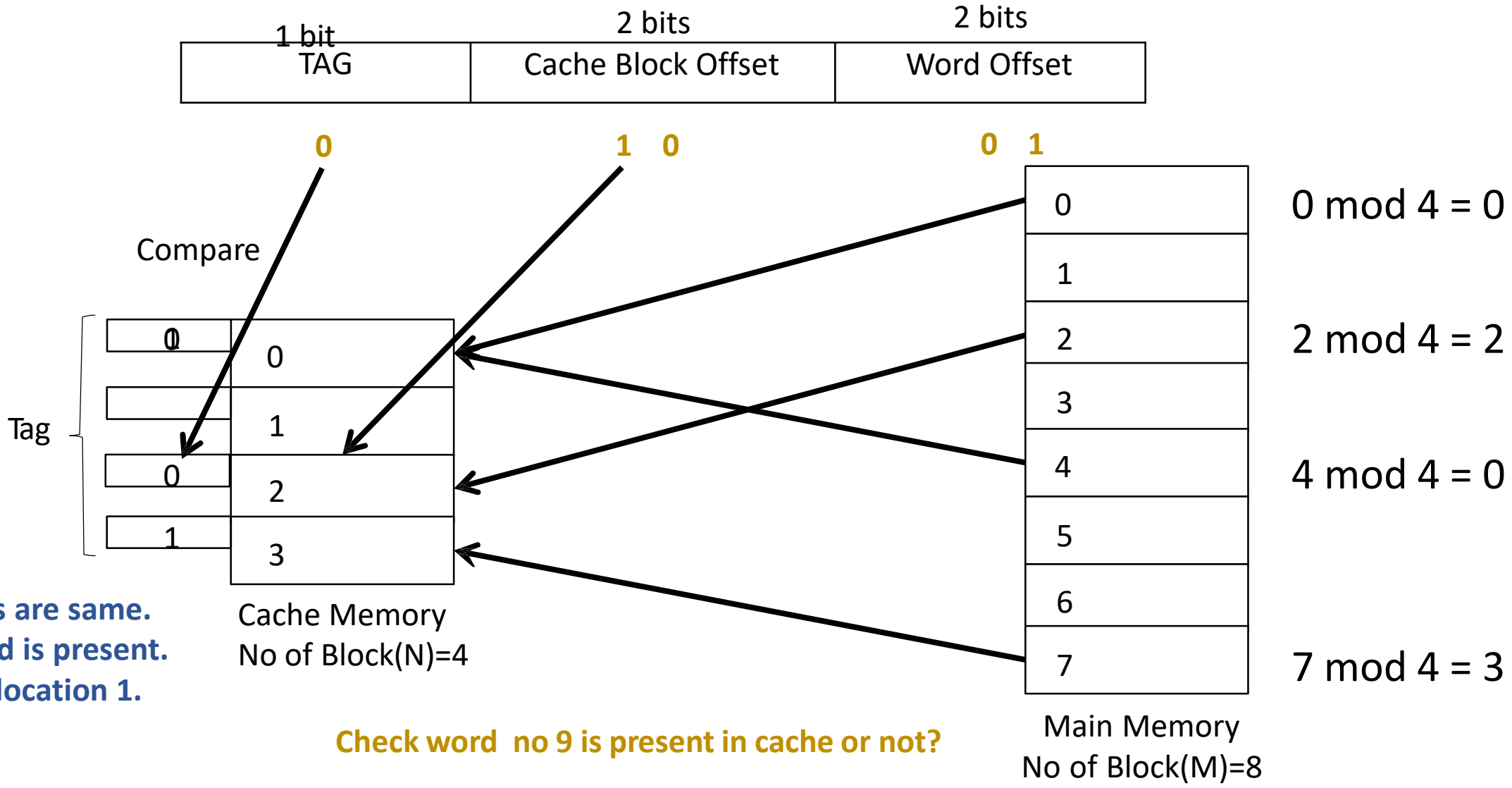
32 words in Main Memory
16 words in Cache
Memory Block size= 4 words

Physical Address= 5 bits



MM Blocks That Can be Placed in Block 0 of CM are {0,4} MM Blocks That Can be Placed in Block 1 of CM are {1,5} MM Blocks That Can be Placed in Block 2 of CM are {2,6} MM Blocks That Can be Placed in Block 3 of CM are {3,7}

Direct Mapping



Tag bits are same.
So word is present.
It is at location 1.

Check word no 9 is present in cache or not?

Physical Address **0 1 0 0 1**

Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for cache offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Direct Mapping

PA= 9 Bits

3 bit	2 bits	4 bits
TAG	Cache Block Offset	Word Offset

Tag

100	16
00	
010	9
01	
110	26
10	
001	
	7
11	

Cache Memory
No of Block(N)=4

10000	16
01001	9
11010	26
00111	7

Main Memory
0
1
2
3
4
:
30
31

Main Memory
No of Block(M)=32

Check word is present in cache or not?

word no 237
PA: 011 10 1101 ✖

TAG is Different in 10

word no 400
PA: 110 01 0000 ✖

TAG is Different in 01

word no 145
PA: 010 01 0001

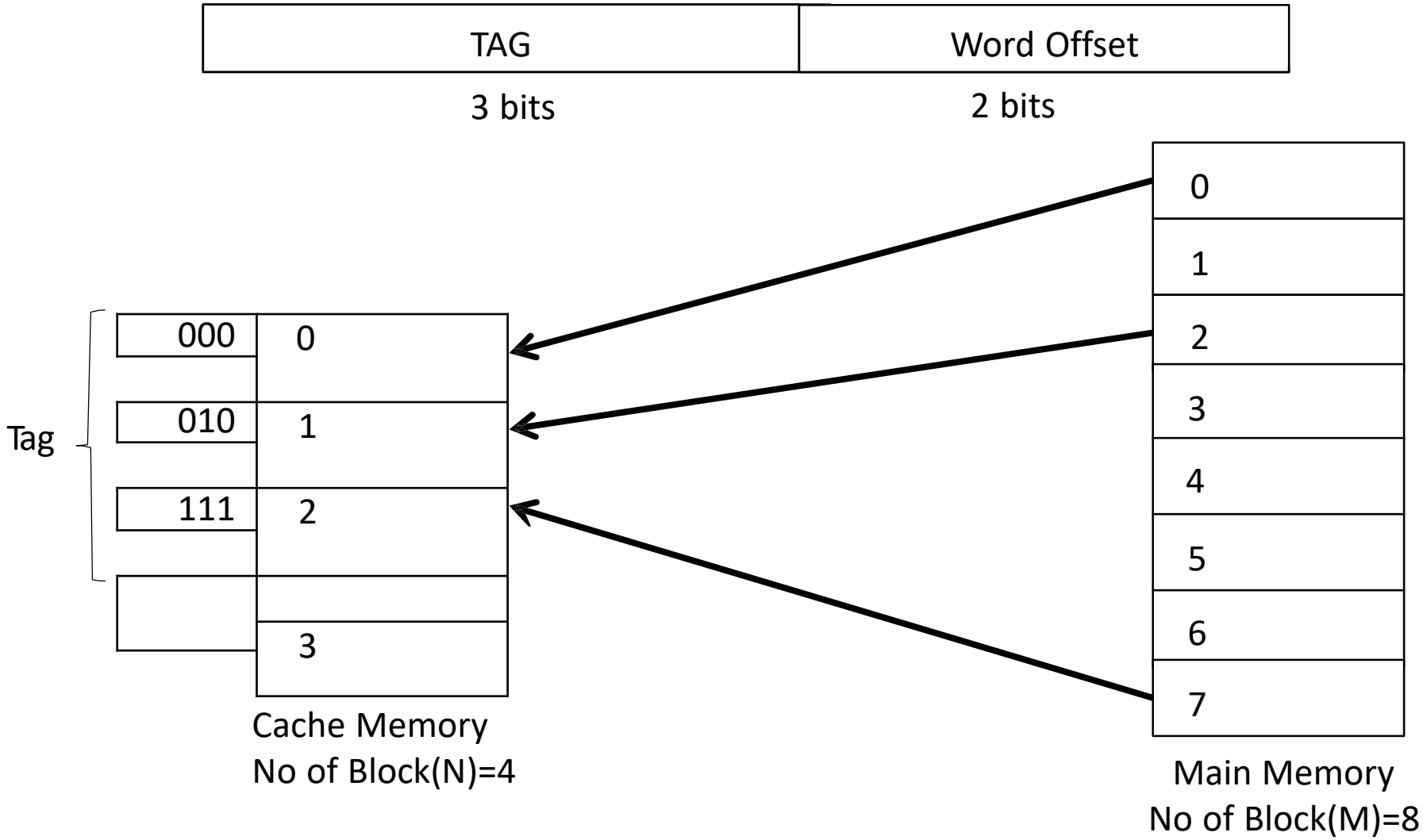
TAG is same in 01 **word is at 0001 in that block**

Limitation of Direct Mapping

- Conflict penalties:
  - Present of one block abstracting the incoming block inspite of the cache is free
- MM Block ref: 0, 4, 0, 4, 8, 0, 4, 12, 4, 12

Associative Mapping

Any Block of main memory can be placed any where in the cache



Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for cache offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Set Associative Mapping

- Advantage of Direct Mapping and Associative Mapping.

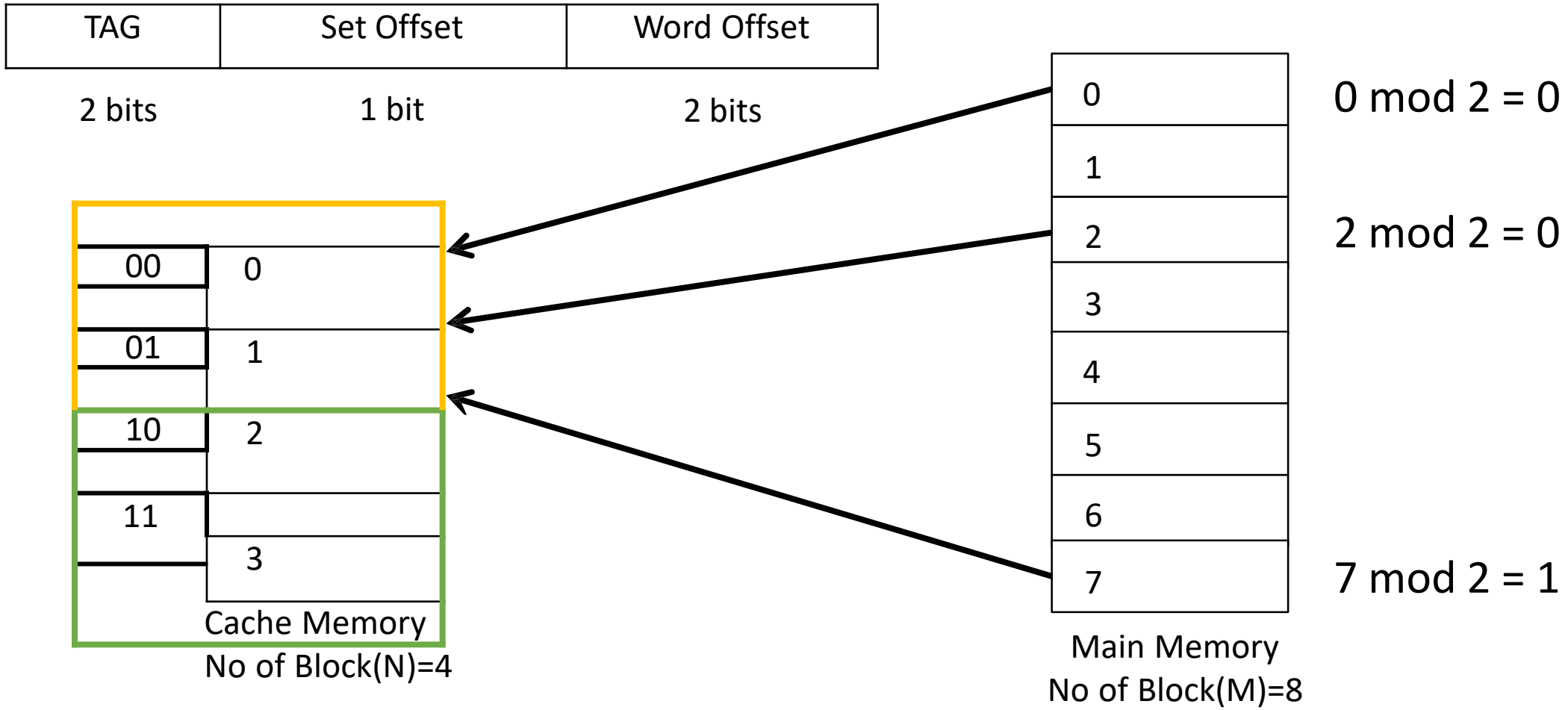
- Total number of sets in k-way set associative is:

$$S = \frac{N\ (Total\ Blocks\ in\ Cache)}{K\ (Blocks\ per\ set)}$$

- J^{th} Block of MM has to be placed in (J^{th} mod S)th Set. Within the set it can be placed anywhere.

2 Way Set Associative Mapping

$$S = \frac{4}{2} = 2 \text{ Sets}$$



Example

- MM= 512 words
- CM = 64 words
- Block size= 16 words
- 2-way Set Associative

1. How many Bits for Physical Address?
2. How many TAG Bits are required?
3. How many Bits are required for set offset?
4. How many Bits are required for word offset?
5. If MM blocks 16, 9, 26, 7 are present in CM than show Tag of each cache.
6. Now Check on above cache representation that word 237 and 400 is Present? and How?

Question

Consider 1MB cache and 4 GB MM are partitioned into 64KB blocks.

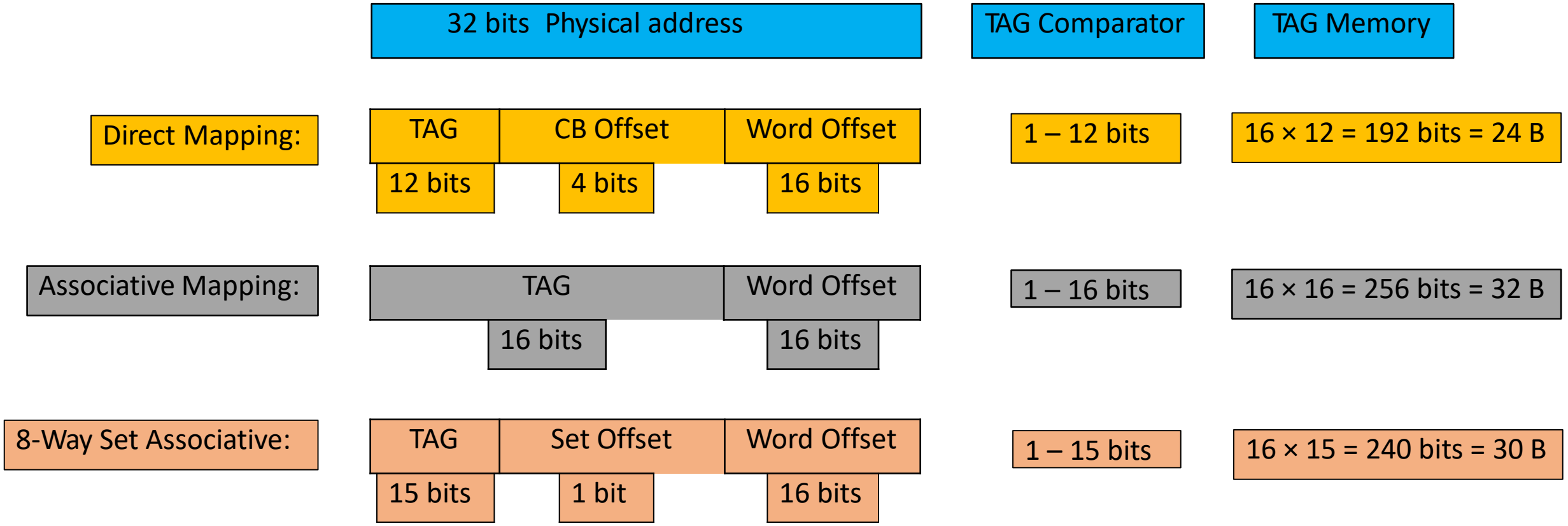
1. How many bits are required for Physical address if word size is 1B?

2. How many block are there in MM and CM?

3. How many TAG bits are required for
    1. Direct Mapping
    2. Associate Mapping
    3. 8-Way set Associative Mapping

4. How many TAG memory in Bytes are required for each case?
   TAG memory size = no. of block in cache memory * no. of tag bits
   TAG Comparator = no. of bits in tag

Answer


32 bits Physical address
TAG Comparator
TAG Memory
Direct Mapping:
TAG
CB Offset
Word Offset
12 bits
4 bits
16 bits
1 – 12 bits
16 × 12 = 192 bits = 24 B
Associative Mapping:
TAG
Word Offset
16 bits
16 bits
1 – 16 bits
16 × 16 = 256 bits = 32 B
8-Way Set Associative:
TAG
Set Offset
Word Offset
15 bits
1 bit
16 bits
1 – 15 bits
16 × 15 = 240 bits = 30 B

Observation

- What about 1 way set associative?
  - It is similar to direct mapping.
  - Because each set contains 1 block. To select set we use modulo operation. So, It will tell you in which set you have to go and there is one block only in that, which is similar to direct mapping.
- What about N-way set associative?
  - It is similar to Associative Mapping.
  - As set size is equal to total number of cache block, there is only 1 set.
  - In set you can put your block any where which is similar to associative mapping.

Writing to main memory

Write-through: When data is updated, it is written to both the cache and the back-end storage. This mode is easy for operation but is slow in data writing because data has to be written to both the cache and the storage.

Write-back: When data is updated, it is written only to the cache. The modified data is written to the back-end storage only when data is removed from the cache. This mode has fast data write speed but data will be lost if a power failure occurs before the updated data is written to the storage.

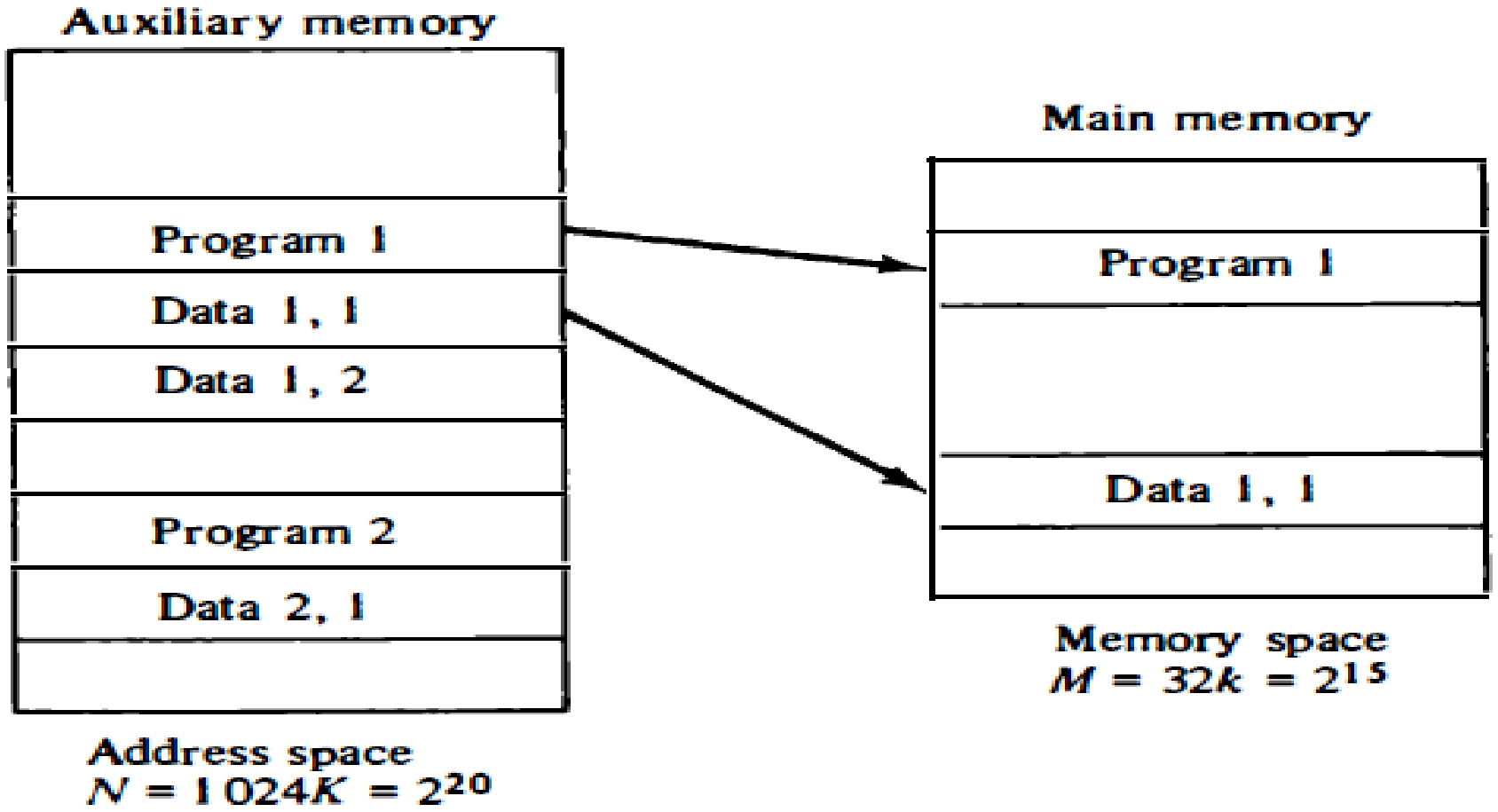
Cache Initialization

- **Valid bit** - A bit of information that indicates whether the data in a block is valid (1) or not (0).

Virtual memory

- Give the programmer the illusion that the system has a very large memory, even though the computer has a relatively small main memory
- An Address used by a programmer will be called a **Virtual Address**, and the set of such addresses called the **Address Space**.
- An Address in Main Memory is called **a Location** or **Physical Address.** The set of such locations is called the **Memory Space.**
- Consider Main Memory is 32K Words = 15 Physical Address line
- Auxiliary Memory capacity = 1024K
- Address Space, N = 1024K
- Memory Space, M = 32K

Virtual memory



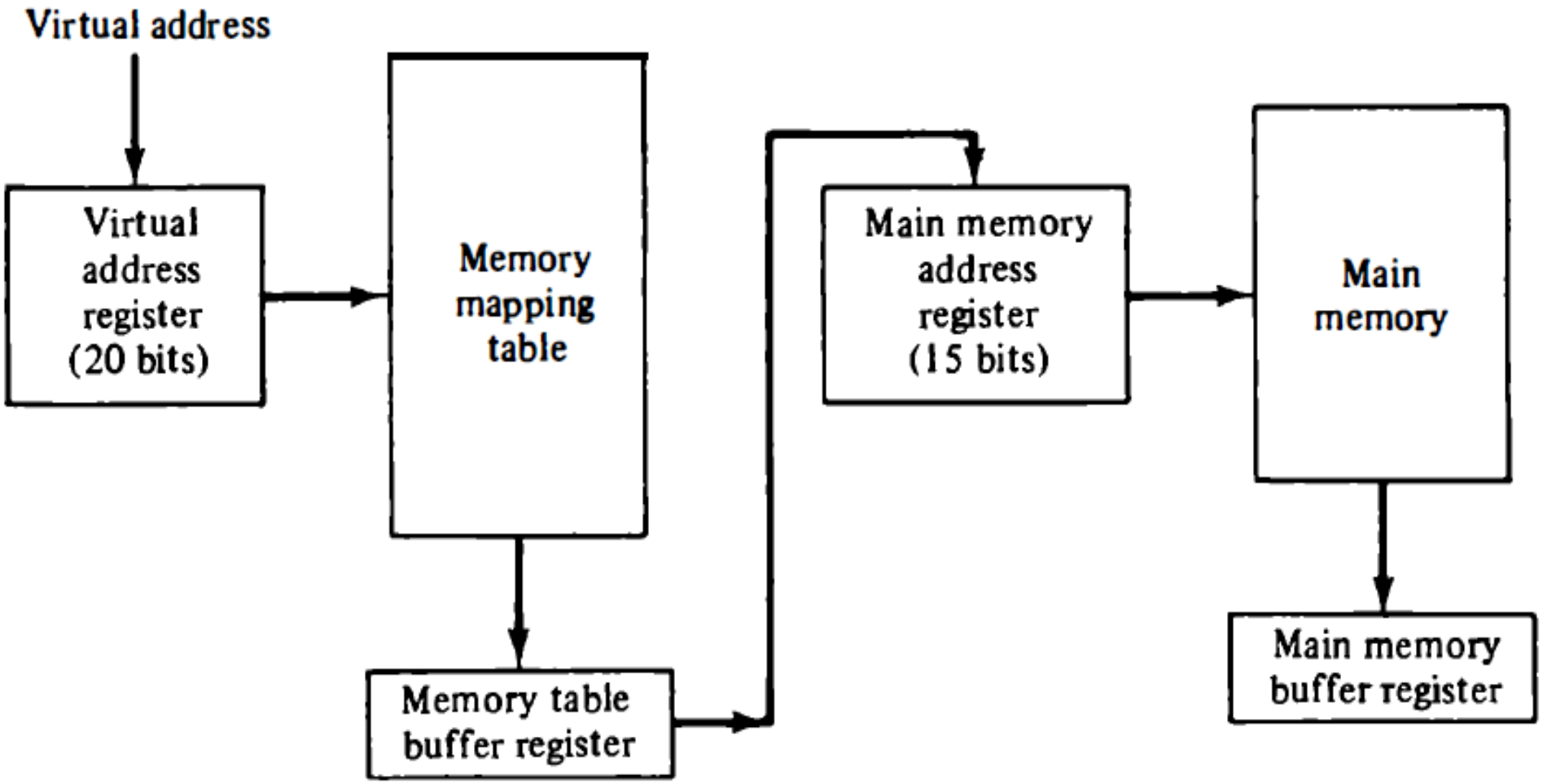
Virtual memory

- The address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits

- A table is then needed, to map a virtual address of 20 bits to a physical address of 15 bits.

- The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU.

Virtual memory



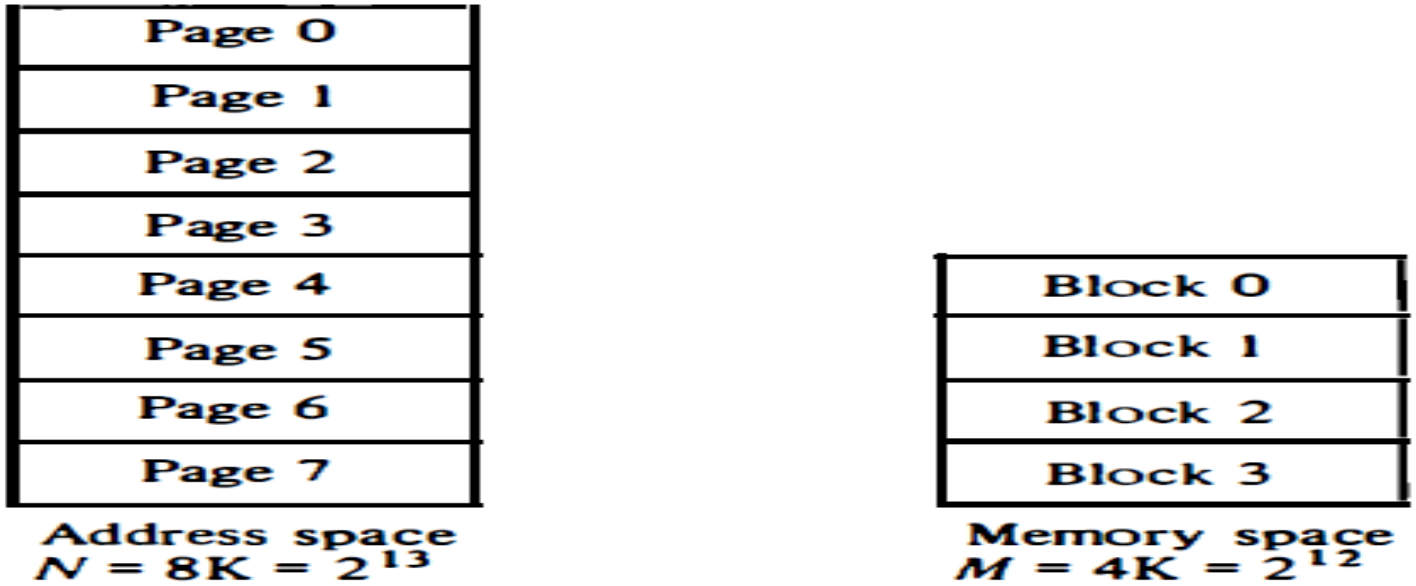
Virtual memory

- **Address Mapping Using Pages:**
- The table implementation of Address Mapping is simplified, if the information in Address Space & Memory Space are divided into groups of fixed size.
- The Physical Memory divided in equal size of groups is called as Blocks.
- The Address Space divided in equal size of groups is called as Pages
- If a Block or Pages consists of 1K then, Address Space divided into 1024 Pages & Memory Space divided into 32 Blocks

Virtual memory

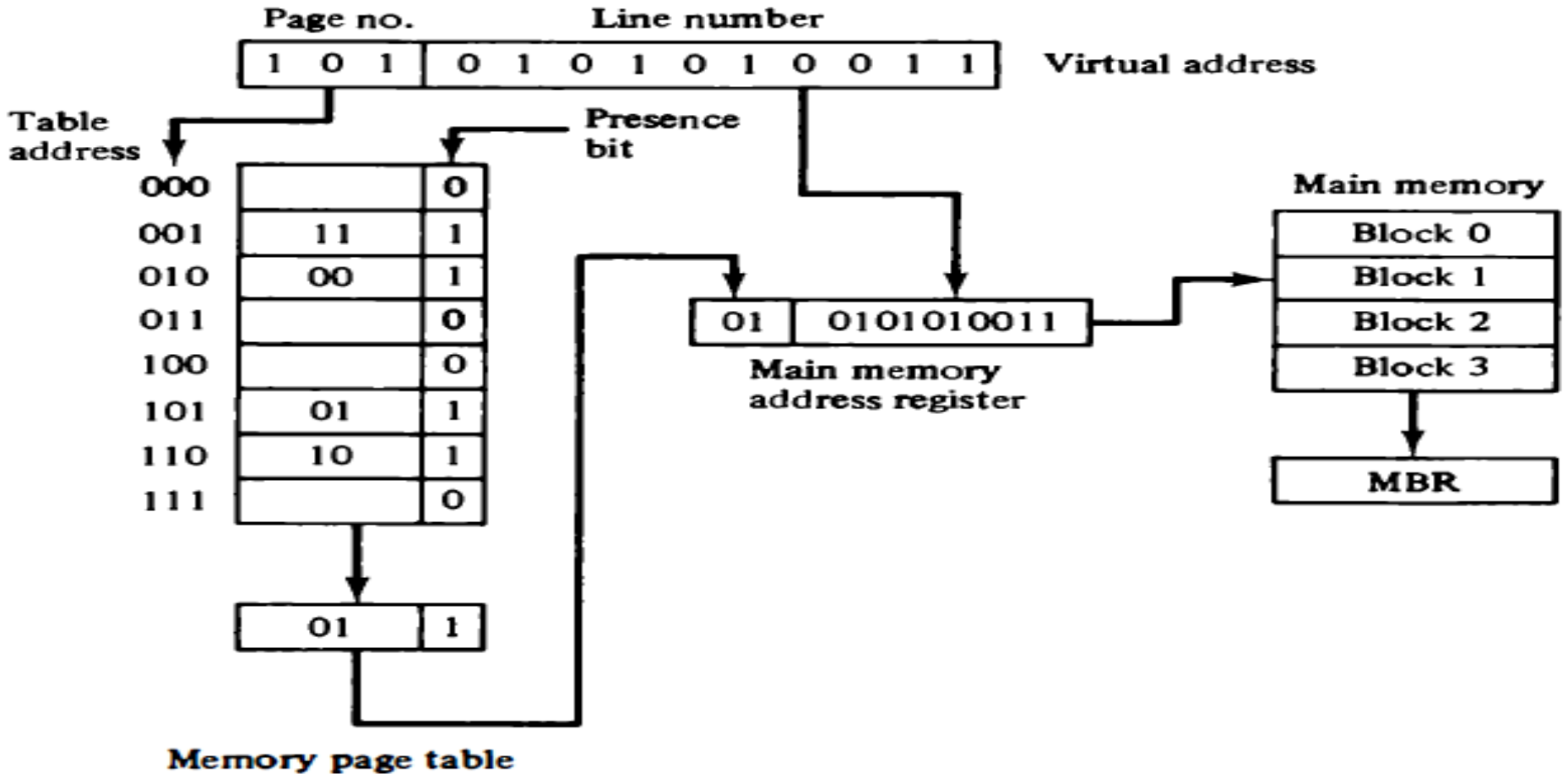
Address Mapping Using Pages:

- Consider a computer a with address space =8K and memory space=4K
- Virtual address has 13 bits. Since each page consists of 2^{10}=1024 words, high-order 3 bits will specify one of 8 pages and low-order 10 bits gives the line address within the pages.



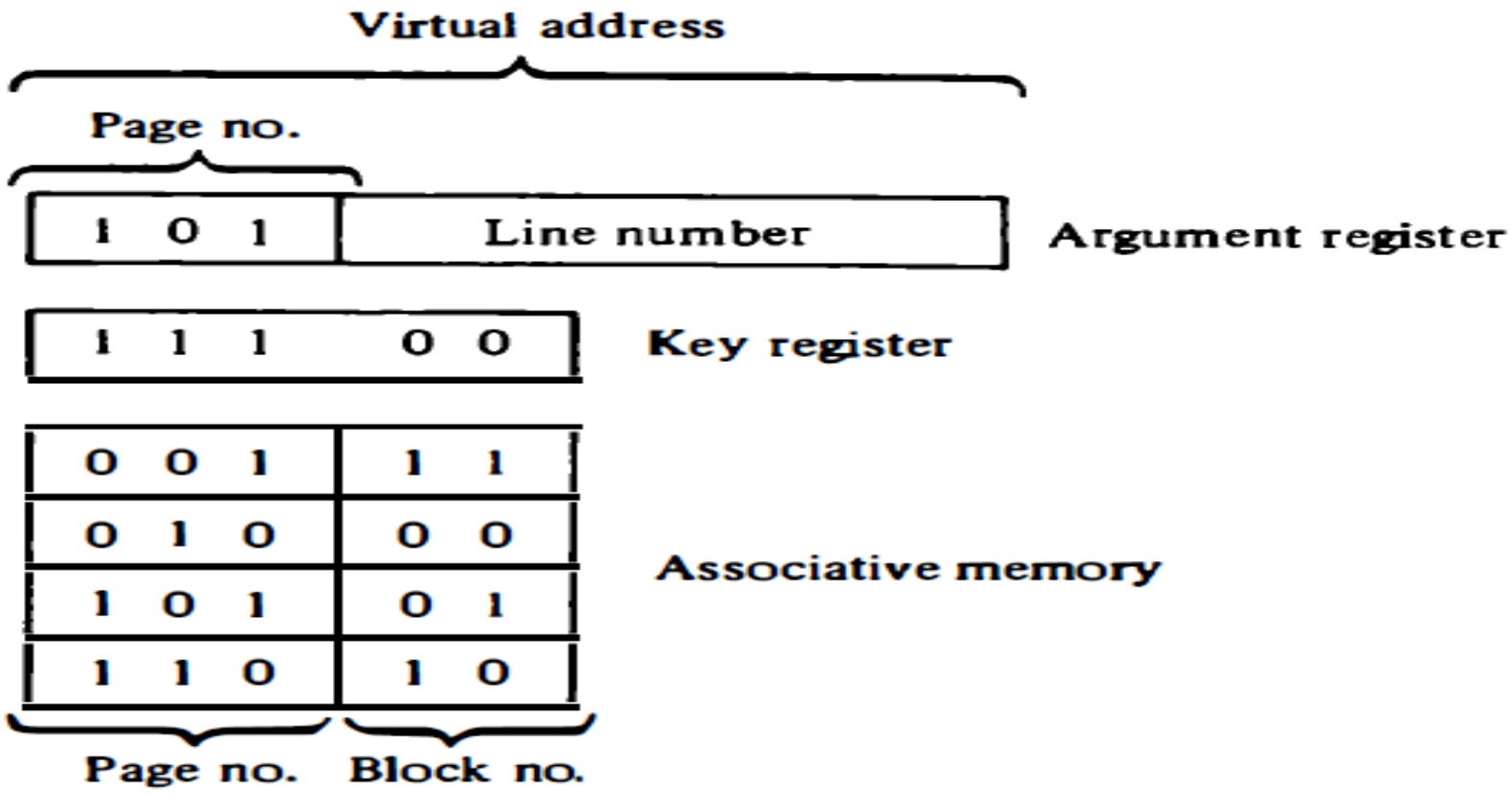
Virtual memory

Address Mapping Using Pages:



Virtual memory

Associative Memory Page Table:



Virtual memory

Page Replacement:

- Unavailability of Page in Cache is called as Page Fault.
- When page fault occurs, the execution of the present program is suspended until the required page is brought into main memory.
- Since loading a page from auxiliary memory to main memory is basically an I/O operation, the operating system assigns this task to the I/O processor.
- In the meantime, control is transferred to the next program in memory that is waiting to be processed in the CPU
- Later, when the memory block has been assigned and the transfer completed, the original program can resume its operation.


Any
Questions
?